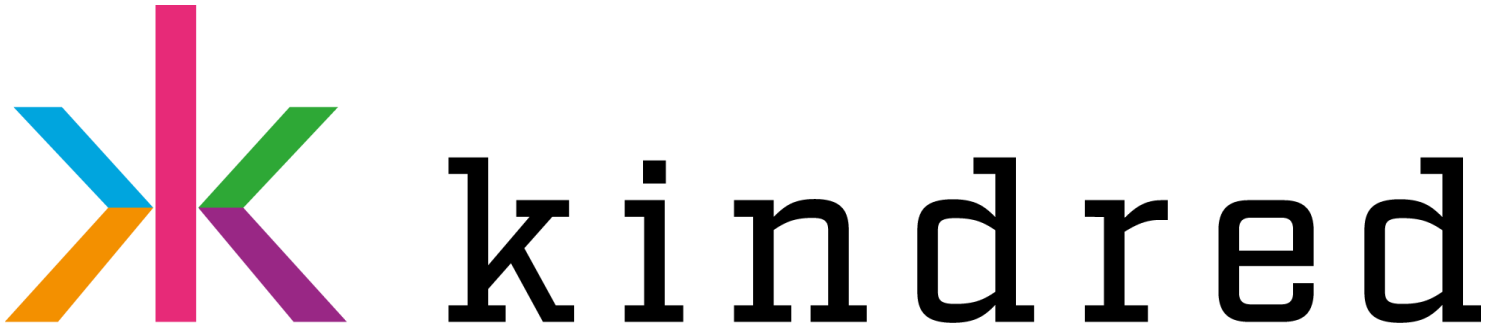
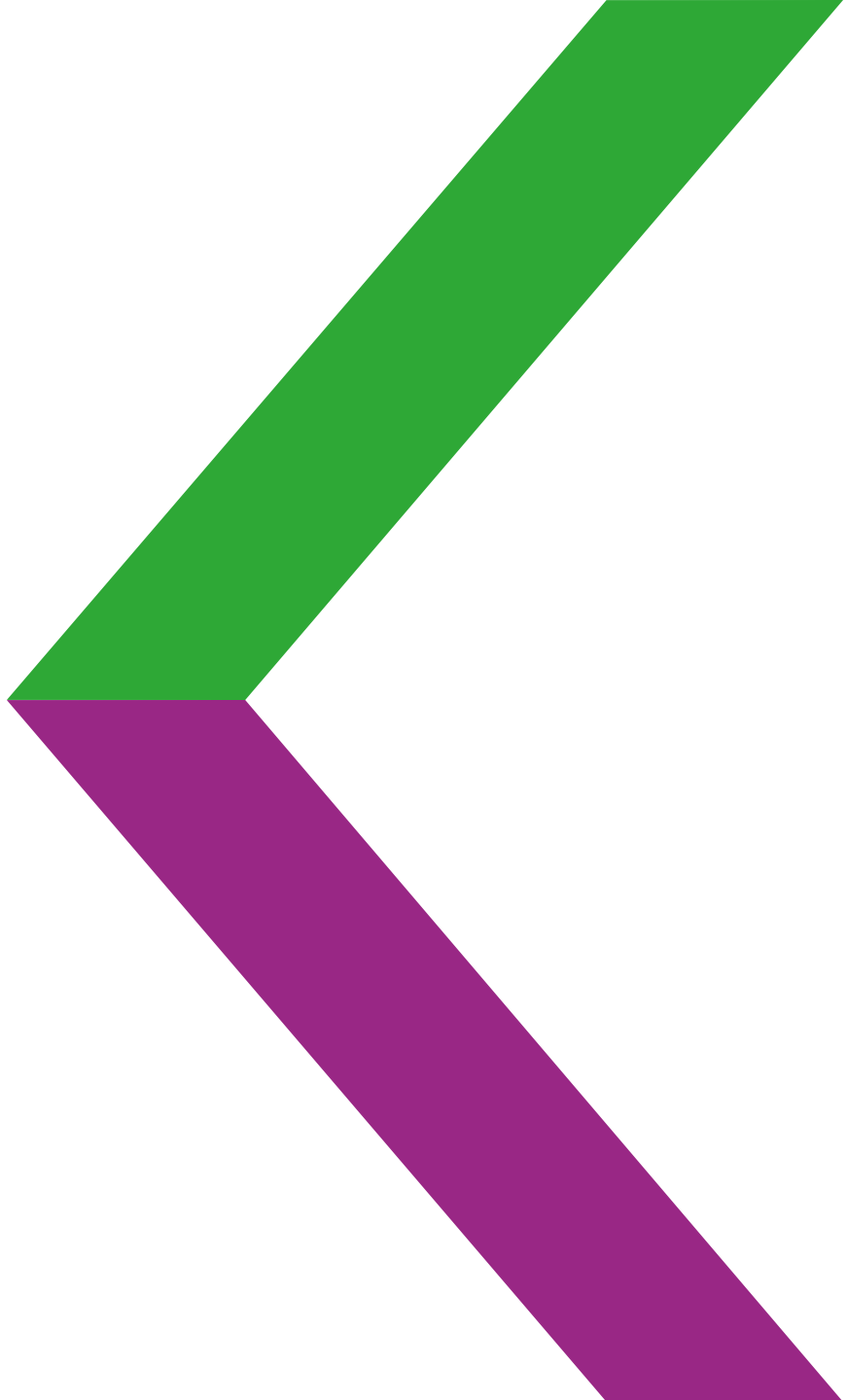


Nolla Utanförskapet: FE JS

Kindred United

Agenda

- JavaScript
- - Examples
- React (brief look into)
- Exercises
- Useful links
- Questions?
-Home assignment



© Kindred Group plc 2018

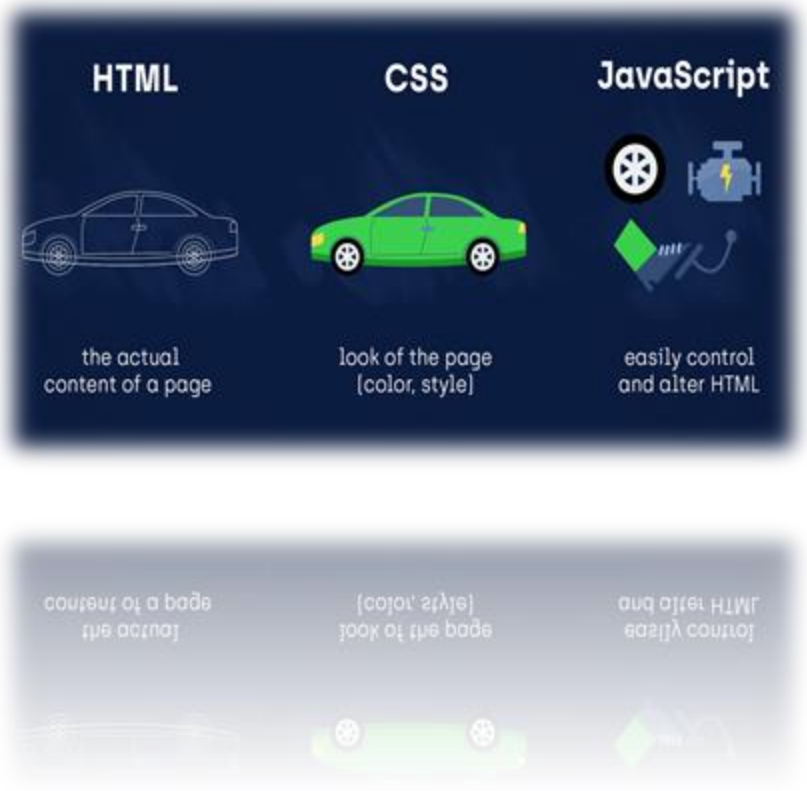
Course Teachers

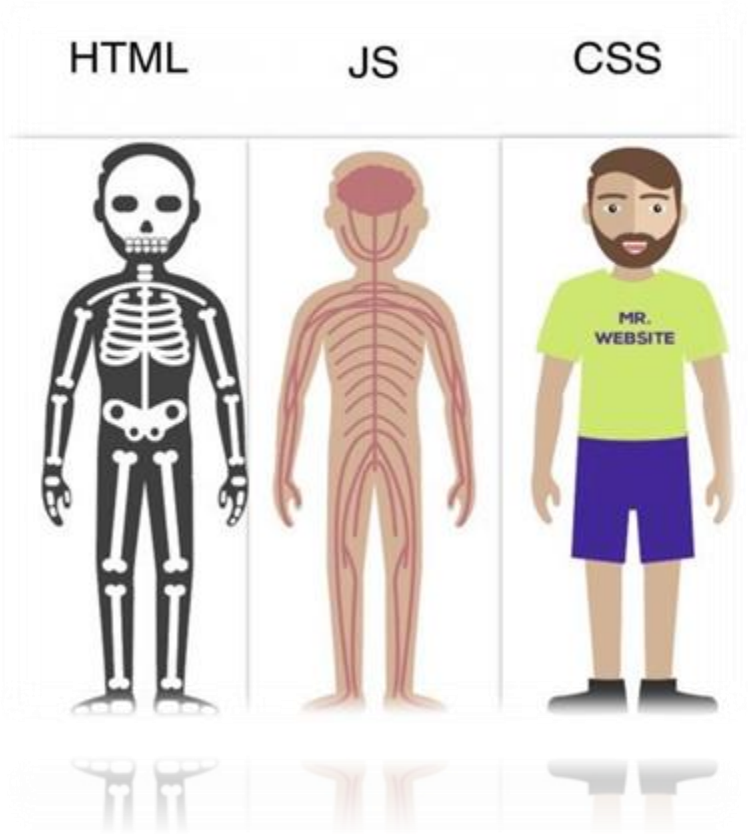
HTML + CSS: (3rd February 2022)	• Elena Insheva • Ishaan Puniani
JavaScript: (8th February 2022)	• Elena Insheva • Erika Boberg
React: (10th February 2022)	• Rodolfo Ortiz

© Kindred Group plc 2021

Classified as General

JS & CSS & HTML





JavaScript to program the behavior of web pages

CSS to specify the layout of web pages

HTML to define the content of web pages

© Kindred Group plc 2021

Classified as General

JavaScript

JavaScript was initially created to *“make web pages alive”*

The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run

JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the JavaScript engine

The browser has an embedded engine sometimes called a “JavaScript virtual machine”

© Kindred Group plc 2021

Classified as General

JavaScript Variables

```
var title = 'Code Life';

let publisher = 'House of Books';

const author = 'Melvin Gray';
```

var – is an old-school variable declaration.

let – is a modern variable declaration.

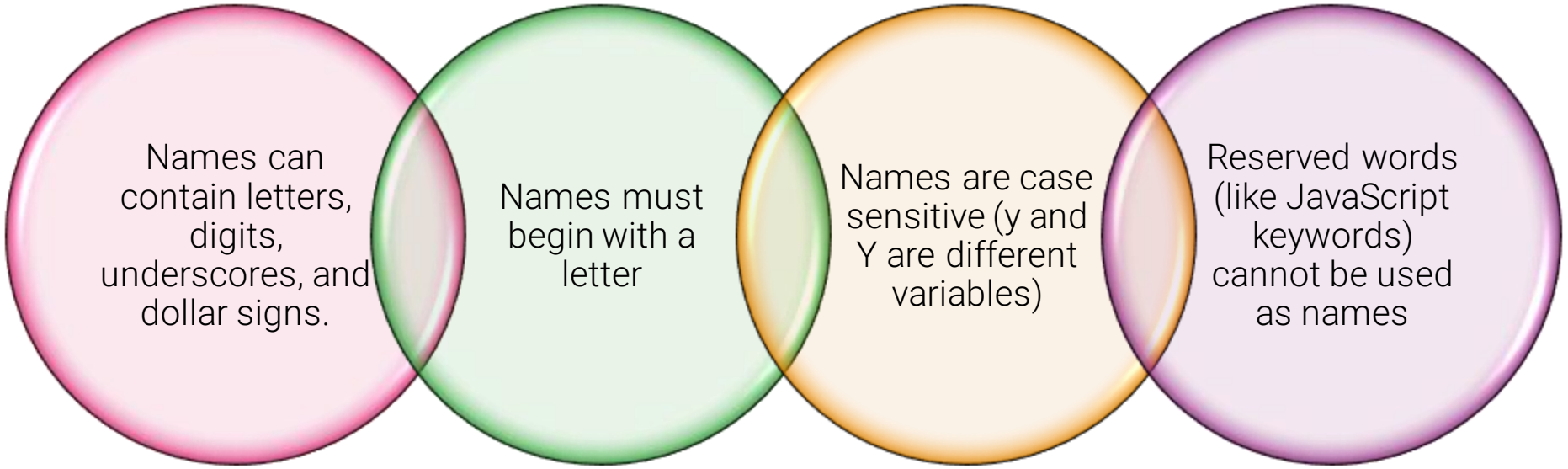
const – is like let, but the value of the variable can't be changed.

© Kindred Group plc 2021

Classified as General

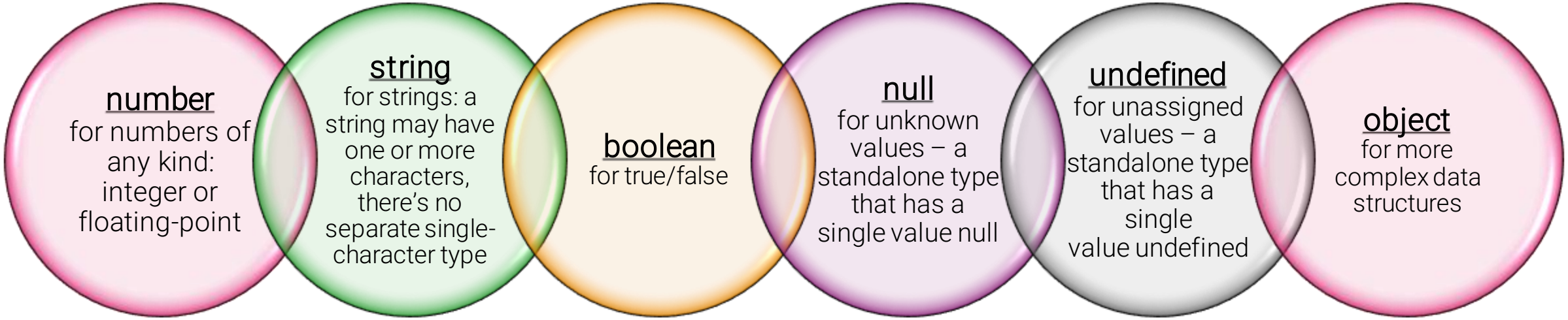
JavaScript Identifiers

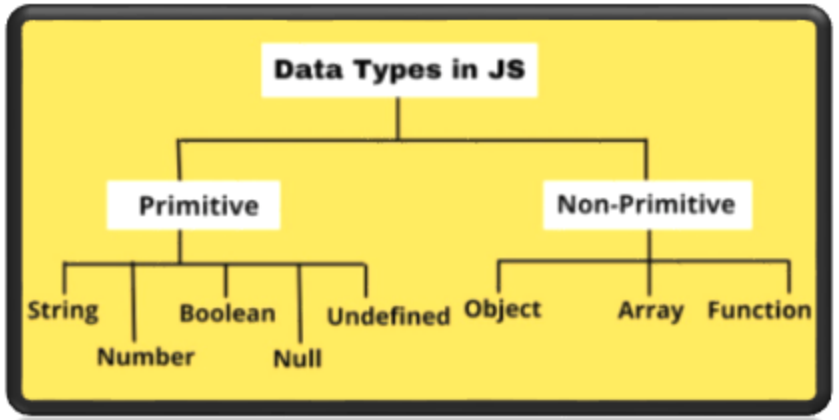
- All JavaScript **variables** must be **identified** with **unique names**.
- These unique names are called **identifiers**.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
- The general rules for constructing names for variables (unique identifiers) are:


Names can contain letters, digits, underscores, and dollar signs.
Names must begin with a letter
Names are case sensitive (y and Y are different variables)
Reserved words (like JavaScript keywords) cannot be used as names

© Kindred Group plc 2021
Classified as General

Data Types in JS


number
for numbers of any kind: integer or floating-point
string
for strings: a string may have one or more characters, there's no separate single-character type
boolean
for true/false
null
for unknown values – a standalone type that has a single value null
undefined
for unassigned values – a standalone type that has a single value undefined
object
for more complex data structures



© Kindred Group plc 2021

Classified as General

JavaScript Objects

- In JavaScript, almost "everything" is an object.
- All JavaScript values, except primitives, are objects.

Example

```
let person = "John Doe";
```

- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

© Kindred Group plc 2021

Classified as General

JavaScript Objects

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Objects</h2>
<p>Creating an object:</p>

<p id="demo"></p>

<script>
const person = {
  firstName : "John",
  lastName  : "Doe",
  age       : 50,
  eyeColor  : "blue"
};

document.getElementById("demo").innerHTML = person.firstName + " " + person.lastName;
</script>

</body>
</html>
```

JavaScript Objects

Creating an object:

John Doe

© Kindred Group plc 2021

Classified as General

JavaScript Functions

- JavaScript functions are defined with the **function** keyword.
- JavaScript functions can be used as values:
- JavaScript functions can be used in expressions:

Example

```
function myFunction(a, b) {
  return a * b;
}
```

Example

```
function myFunction(a, b) {
  return a * b;
}

let x = myFunction(4, 3);
```

Example

```
function myFunction(a, b) {
  return a * b;
}

let x = myFunction(4, 3) * 2;
```

© Kindred Group plc 2021

Classified as General

JavaScript events

onchange - HTML element has been changed

onclick - user clicks an HTML element

onmouseover - user moves the mouse over an HTML element

onmouseout - user moves the mouse away from an HTML element

onkeydown - user pushes a keyboard key

onload - browser has finished loading the page

"onclick"

© Kindred Group plc 2021

Classified as General

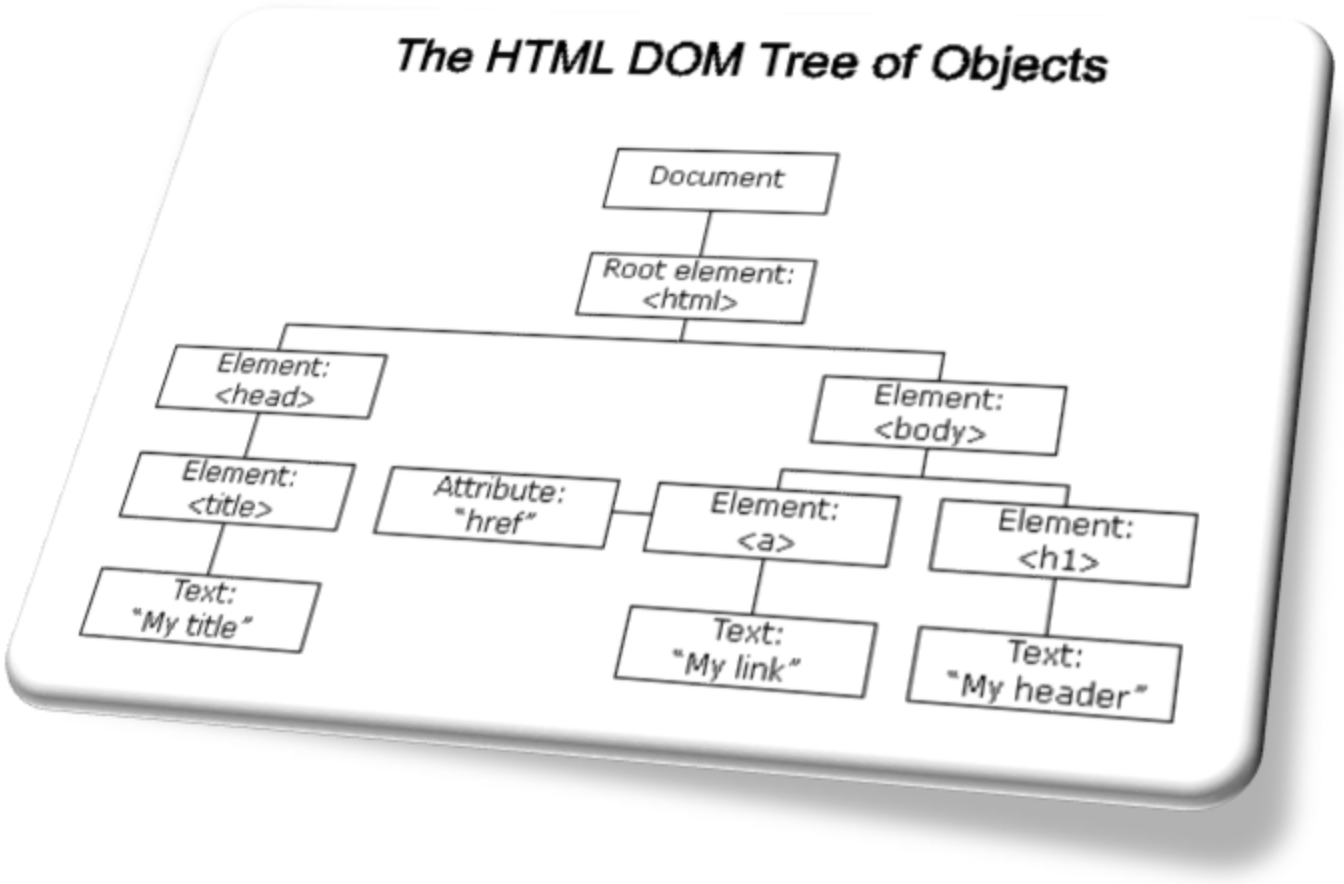
JS can...

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

© Kindred Group plc 2021

Classified as General

The HTML DOM (Document Object Model)



The HTML DOM is a standard **object** model and **programming interface** for HTML

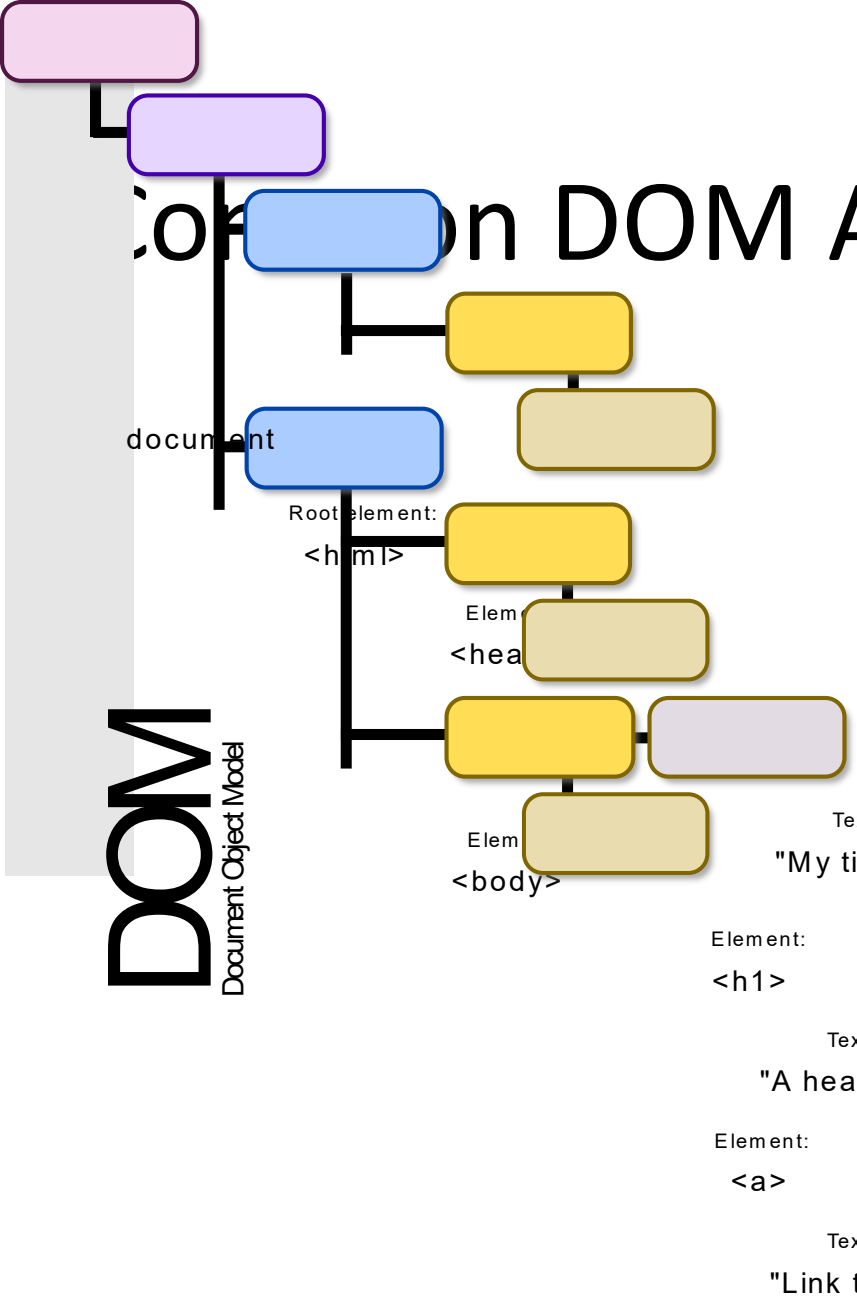
With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

When a web page is loaded, the browser creates a Document Object Model of the page. The **HTML DOM** model is constructed as a tree of **Objects**

© Kindred Group plc 2021

Classified as General

Common DOM APIs



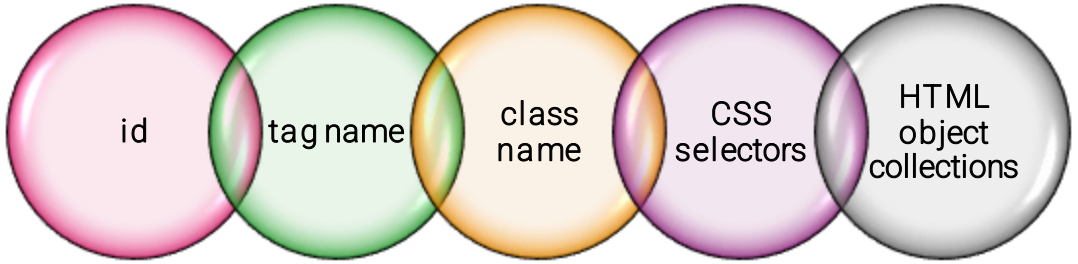
- document.getElementById(id)
- document.getElementsByTagName(name)
- document.createElement(name)
- parentNode.appendChild(node)
- element.innerHTML

© Kindred Group plc 2021

Classified as General

Finding HTML Elements

by:



Example

```
const x = document.forms["frm1"];
let text = "";
for (let i = 0; i < x.length; i++) {
  text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Example

```
const x = document.querySelectorAll("p.intro");
```

Example

```
const element = document.getElementById("intro");
```

Example

```
const x = document.getElementsByClassName("intro");
```

Example

```
const element = document.getElementsByTagName("p");
```

© Kindred Group plc 2021

Classified as General

Changing HTML Elements

Property	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Example

```
<html>
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>

</body>
</html>
```

Example

```
<!DOCTYPE html>
<html>
<body>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

© Kindred Group plc 2021

Classified as General

Adding and Deleting HTML Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Example

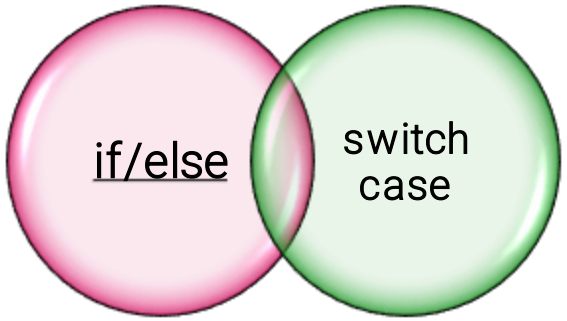
```
<!DOCTYPE html>
<html>
<body>

<p>Bla bla bla</p>

<script>
document.write(Date());
</script>

<p>Bla bla bla</p>

</body>
</html>
```

© Kindred Group plc 2021

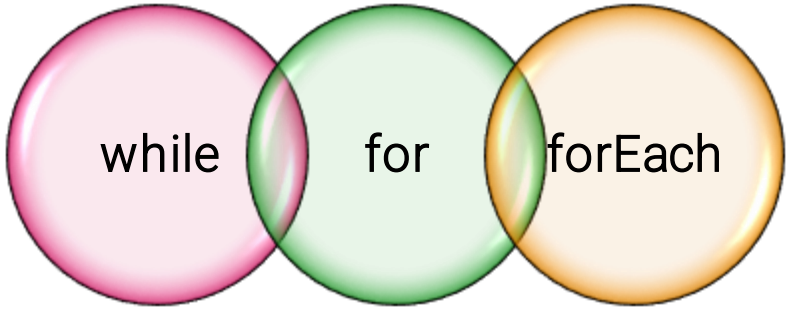
Classified as General

Conditional Statement, Looping and Array utility

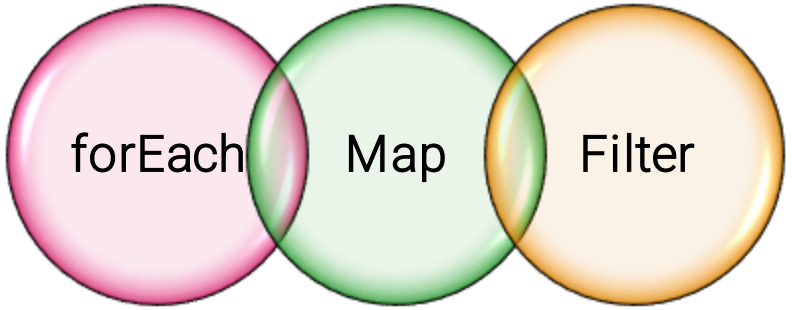
Conditional Statements



Looping



Array utility



© Kindred Group plc 2021

Classified as General

Conditional Statements

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript if .. else</h2>

<p>A time-based greeting:</p>

<p id="demo"></p>

<script>
const time = new Date().getHours();
let greeting;
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
document.getElementById("demo").innerHTML
= greeting;
</script>

</body>
</html>
```

JavaScript if .. else

A time-based greeting:

Good day

Use if to specify a block of code to be executed, if a specified condition is true

Use else to specify a block of code to be executed, if the same condition is false

Use else if to specify a new condition to test, if the first condition is false

Use switch to specify many alternative blocks of code to be executed

© Kindred Group plc 2021

Classified as General

Switch Statement

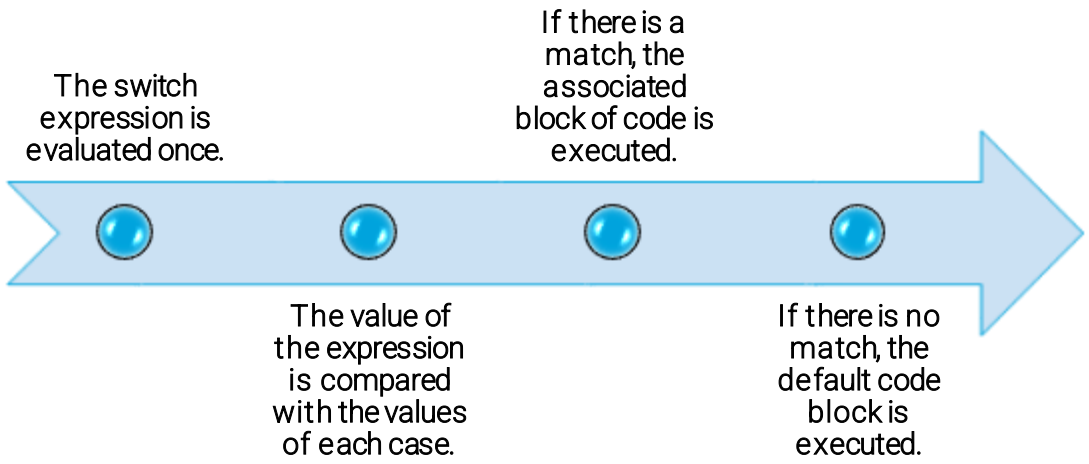
```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript switch</h2>

<p id="demo"></p>

<script>
let day;
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case  6:
    day = "Saturday";
}
document.getElementById("demo").innerHTML = "Today is " + day;
</script>

</body>
</html>
```

JavaScript switch

Today is Thursday

The <u>switch</u> statement is used to perform different actions based on different conditions

Use the <u>switch</u> statement to select one of many code blocks to be executed


The switch expression is evaluated once.
The value of the expression is compared with the values of each case.
If there is a match, the associated block of code is executed.
If there is no match, the default code block is executed.

© Kindred Group plc 2021

Classified as General

Looping

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For Loop</h2>

<p id="demo"></p>

<script>
const cars = ["BMW", "Volvo", "Saab", "Ford",
"Fiat", "Audi"];

let text = "";
for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}

document.getElementById("demo").innerHTML =
text;
</script>

</body>
</html>
```

JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi

Loops can execute a block of code a number of times

Loops are handy, if you want to run the same code over and over again, each time with a different value

Example: use loops instead of writing repetitions of actions like:

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

© Kindred Group plc 2021

Classified as General

Array utility: forEach

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Arrays</h1>
<h2>The forEach() Method</h2>

<p>forEach() calls a function for each element in
an array:</p>

<p>Compute the sum of the values in an array:</p>

<p id="demo"></p>

<script>
let sum = 0;
const numbers = [65, 44, 12, 4];
numbers.forEach(myFunction);

document.getElementById("demo").innerHTML = sum;

function myFunction(item) {
  sum += item;
}
</script>

</body>
</html>
```

JavaScript Arrays

The forEach() Method

forEach() calls a function for each element in an array:

Compute the sum of the values in an array:

125

The forEach() method calls a function for each element in an array

The forEach() method is not executed for empty elements

Multiply each element:

```
const numbers = [65, 44, 12, 4];
numbers.forEach(myFunction)

function myFunction(item, index, arr) {
  arr[index] = item * 10;
}
```

© Kindred Group plc 2021

Classified as General

Array utility: map

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Arrays</h2>

<p>Multiply every element in the array
with 10:</p>

<p id="demo"></p>

<script>
const numbers = [65, 44, 12, 4];
const newArr = numbers.map(myFunction);

document.getElementById("demo").innerHTML
= newArr;

function myFunction(num) {
  return num * 10;
}
</script>

</body>
</html>
```

JavaScript Arrays

Multiply every element in the array with 10:

650,440,120,40

map() creates a new array from calling a function for every array element

map() calls a function once for each element in an array

map() does not execute the function for empty elements

map() does not change the original array

Return a new array with the square root of all element values:

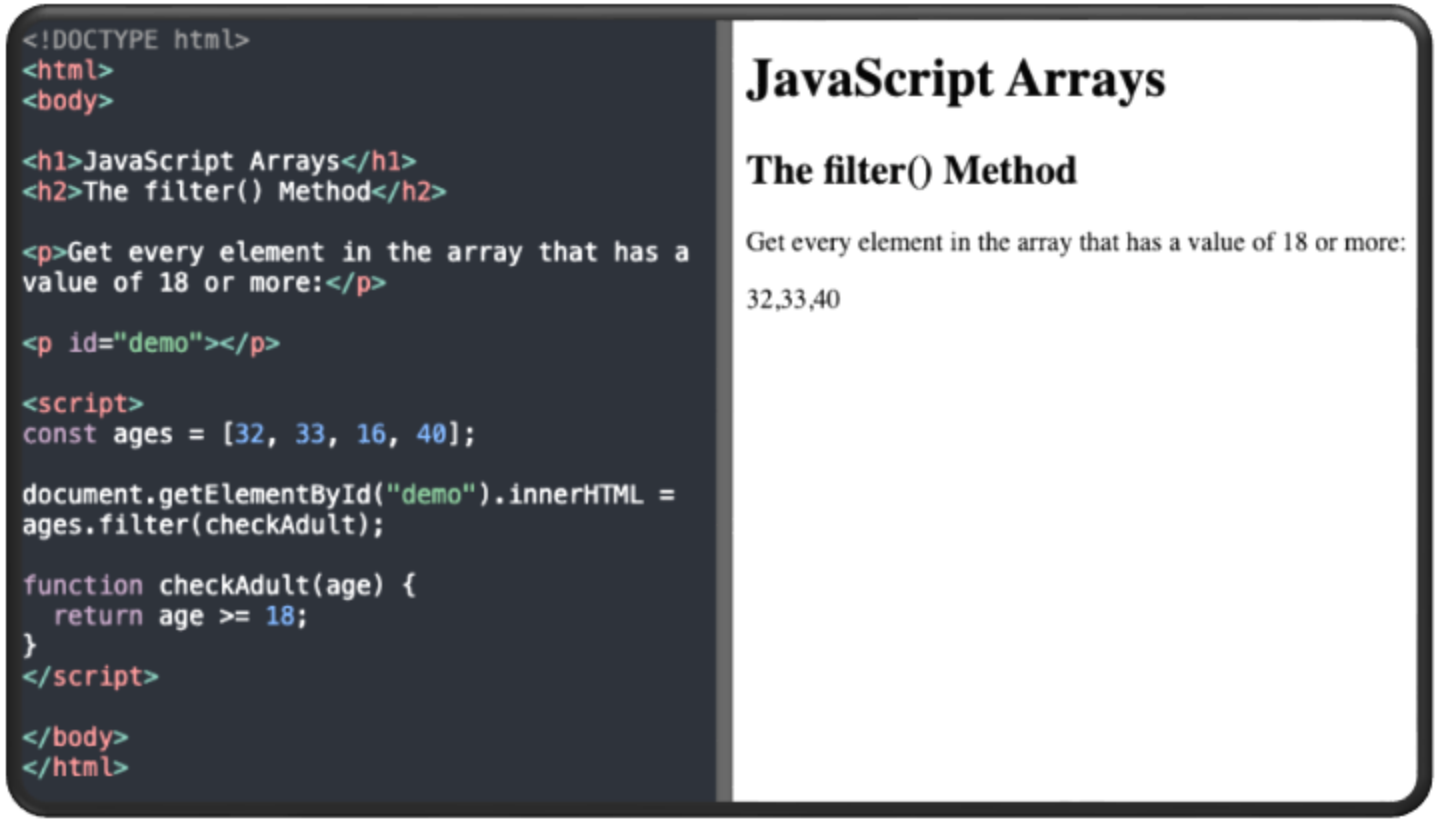
```
const numbers = [4, 9, 16, 25];
const newArr = numbers.map(Math.sqrt)
```

© Kindred Group plc 2021

Classified as General

Array utility: filter

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Arrays</h1>
<h2>The filter() Method</h2>

<p>Get every element in the array that has a
value of 18 or more:</p>

<p id="demo"></p>

<script>
const ages = [32, 33, 16, 40];

document.getElementById("demo").innerHTML =
ages.filter(checkAdult);

function checkAdult(age) {
  return age >= 18;
}
</script>

</body>
</html>
```



The filter() method creates a new array filled with elements that pass a test provided by a function

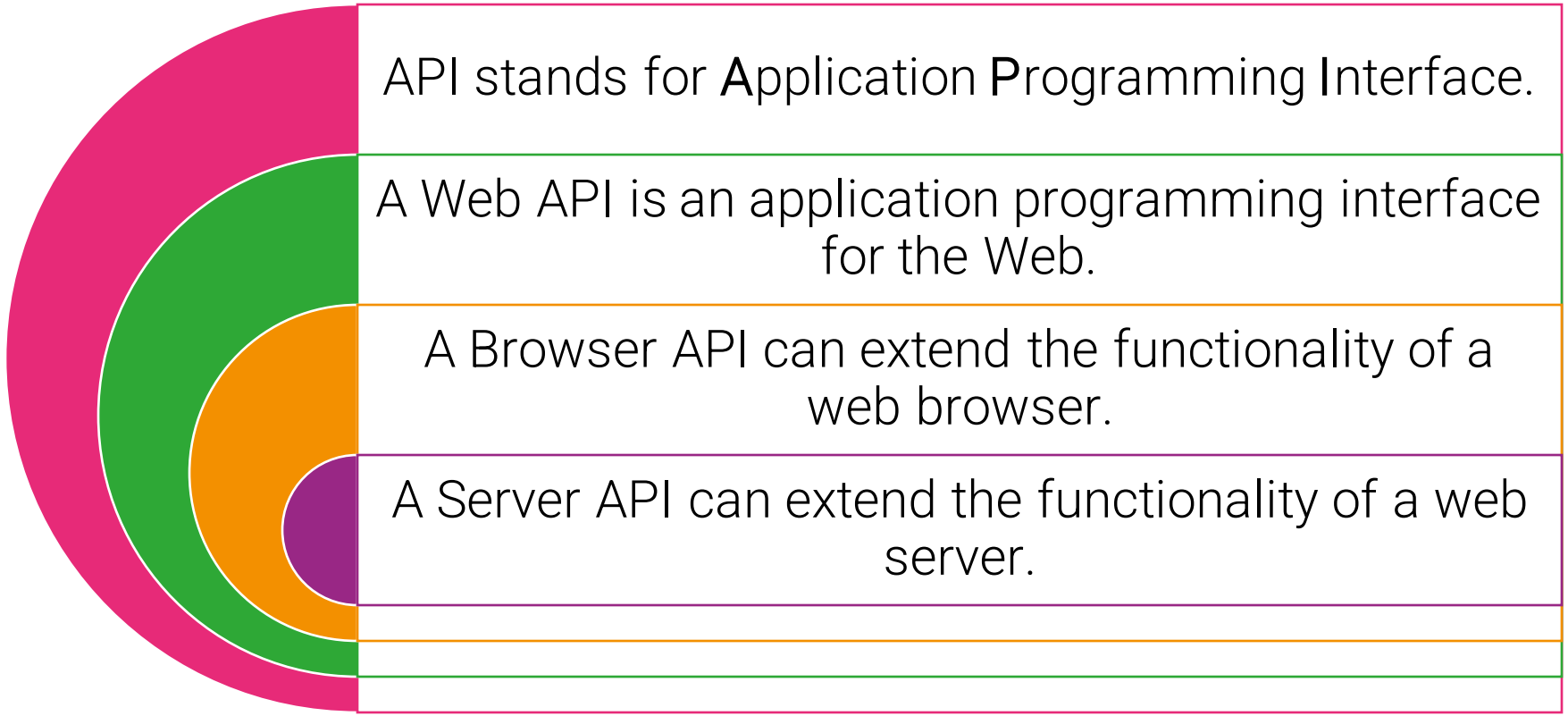
The filter() method does not execute the function for empty elements

The filter() method does not change the original array

© Kindred Group plc 2021

Classified as General

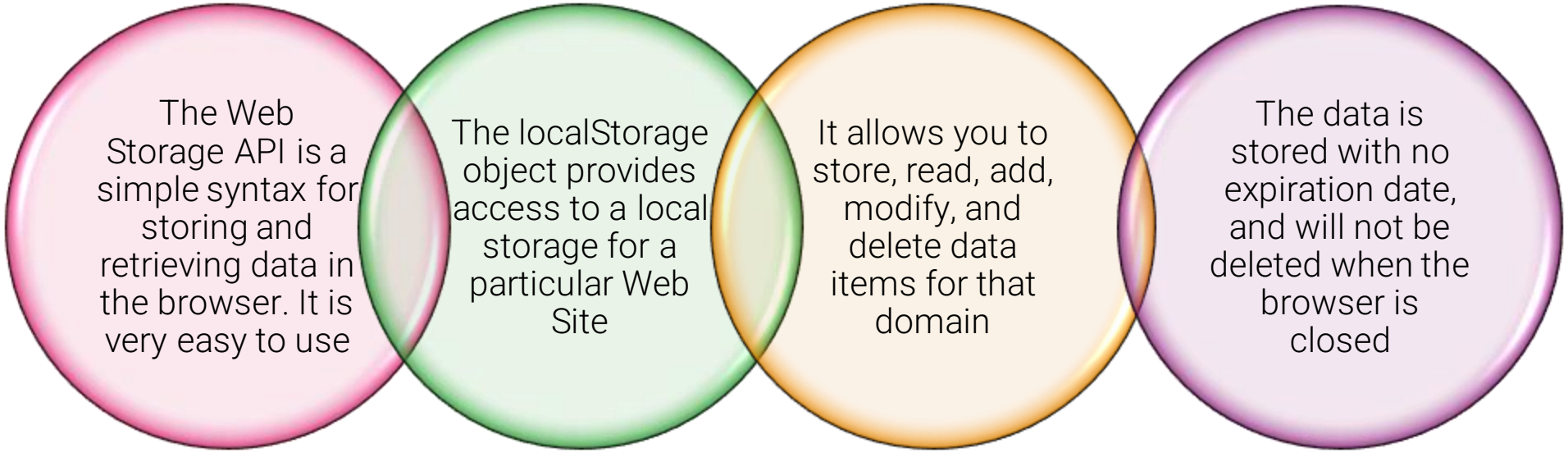
JS Web APIs


API stands for Application Programming Interface.
A Web API is an application programming interface for the Web.
A Browser API can extend the functionality of a web browser.
A Server API can extend the functionality of a web server.

© Kindred Group plc 2021

Classified as General

JS Web Storage API: local storage


The Web Storage API is a simple syntax for storing and retrieving data in the browser. It is very easy to use
The localStorage object provides access to a local storage for a particular Web Site
It allows you to store, read, add, modify, and delete data items for that domain
The data is stored with no expiration date, and will not be deleted when the browser is closed

Example

```
localStorage.setItem("name", "John Doe");
localStorage.getItem("name");
```

© Kindred Group plc 2021

Classified as General

JS Web Storage API: session storage

The sessionStorage object is identical to the localStorage object
The difference is that the sessionStorage object stores data for one session
The data is deleted when the browser is closed

Example

```
sessionStorage.setItem("name", "John Doe");
```

Example

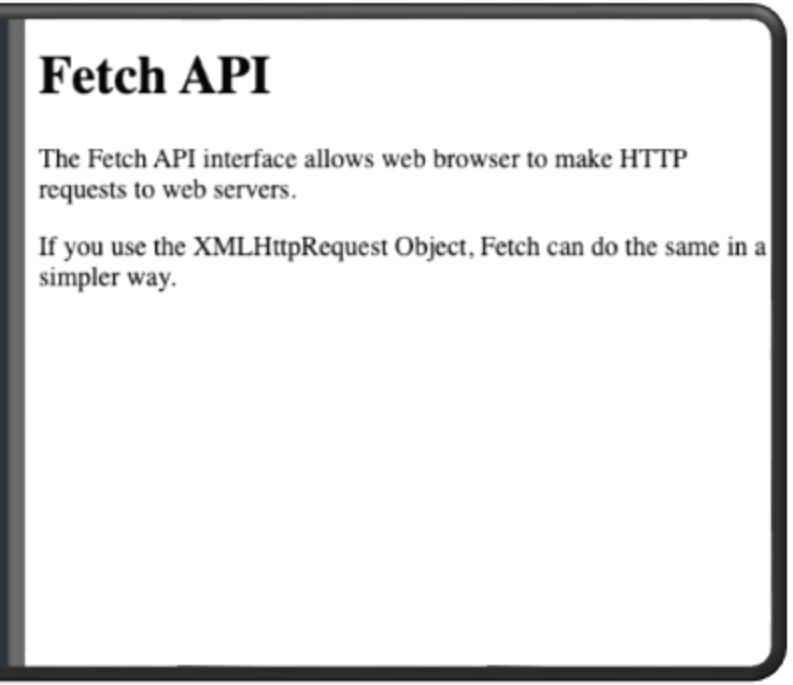
```
sessionStorage.getItem("name");
```

© Kindred Group plc 2021

Classified as General

JS Fetch API

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Fetch a file to change this text.
</p>

<script>
getText("fetch_info.txt");

async function getText(file) {
  let myObject = await fetch(file);
  let myText = await myObject.text();
  document.getElementById("demo").innerHTML =
myText;
}
</script>

</body>
</html>
```


Fetch API
The Fetch API interface allows web browser to make HTTP requests to web servers.
If you use the XMLHttpRequest Object, Fetch can do the same in a simpler way.

The Fetch API interface allows web browser to make HTTP requests to web servers.

© Kindred Group plc 2021

Classified as General

JS & React

React is a JavaScript library for building user interfaces

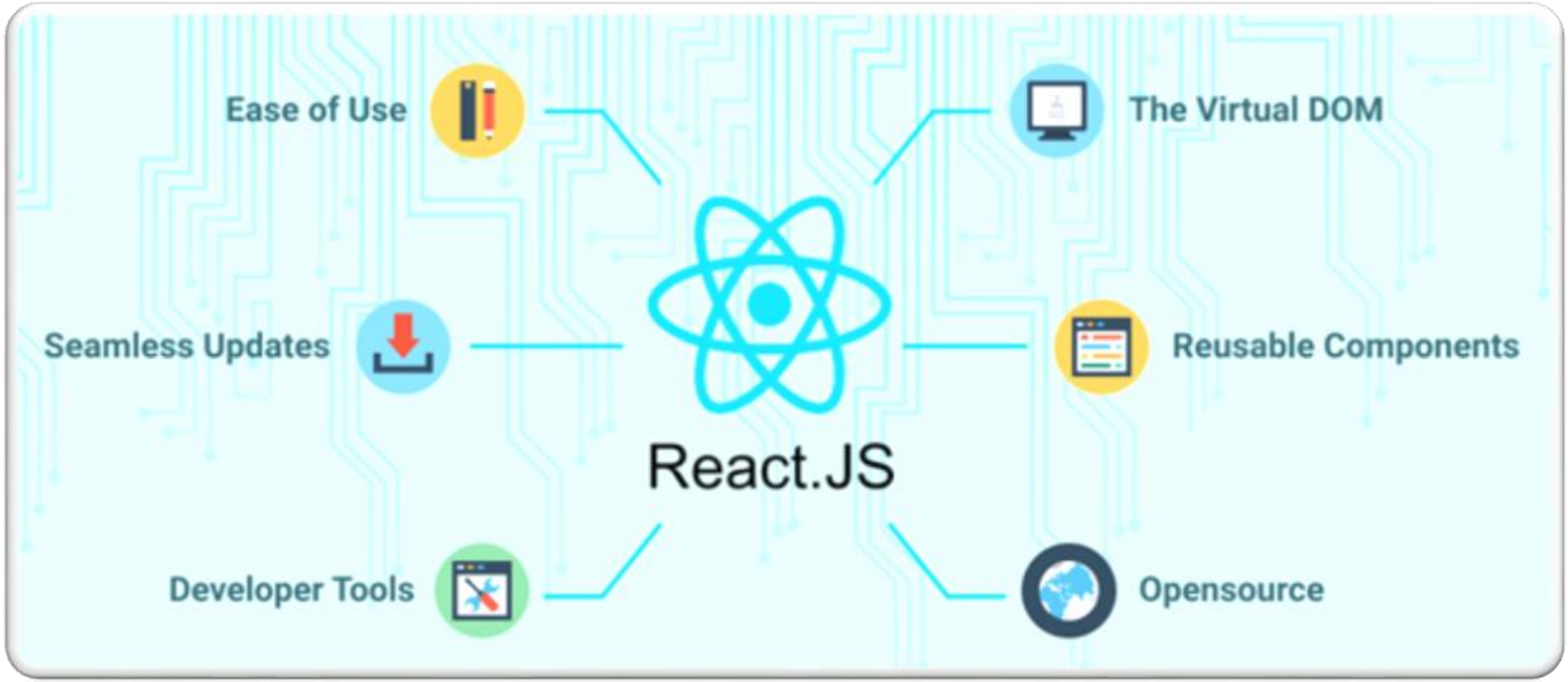
React is used to build single-page applications

React allows us to create reusable UI components

© Kindred Group plc 2021

Classified as General

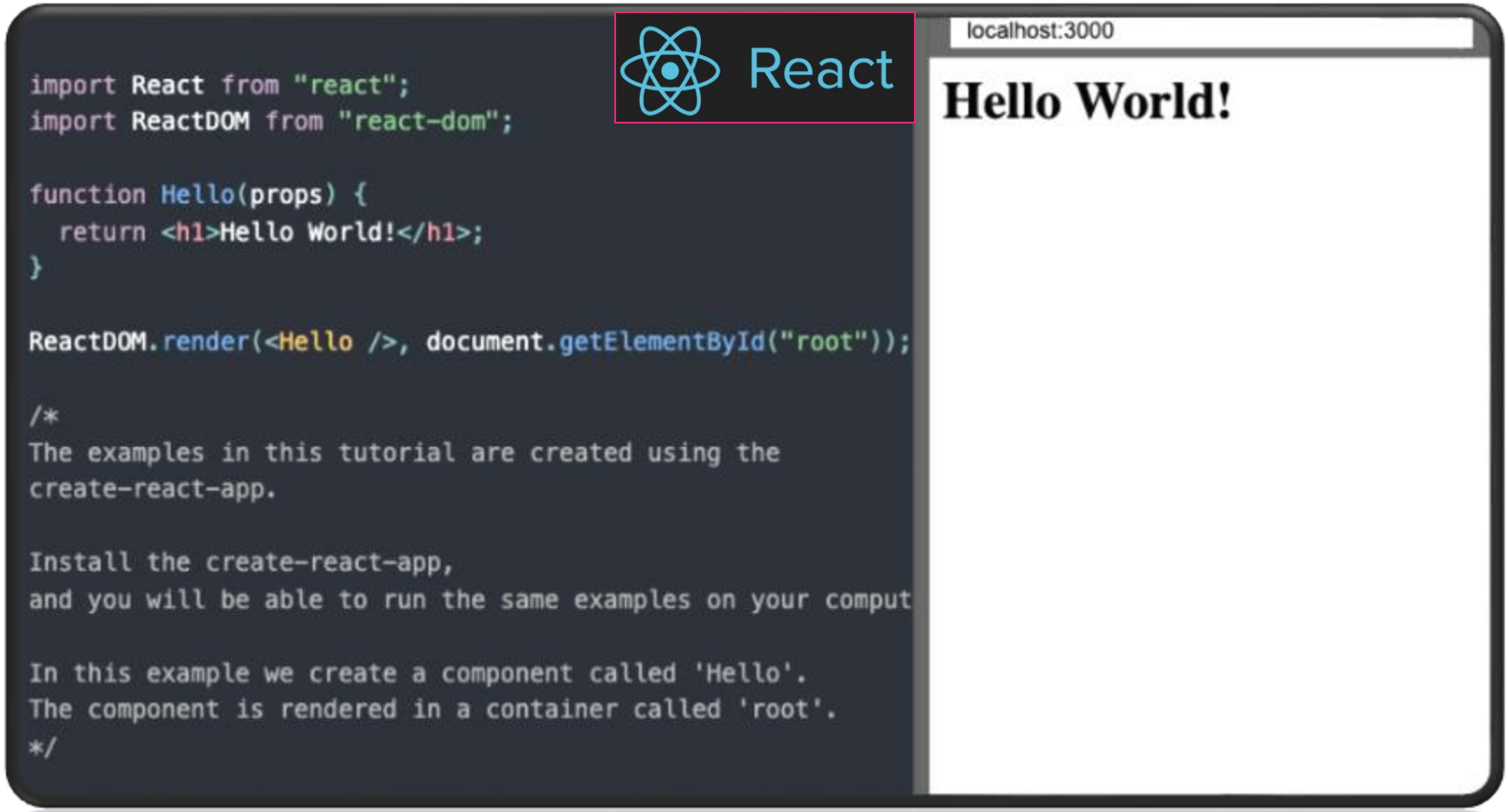
Why React?



© Kindred Group plc 2021

Classified as General

React syntax



React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

React is a tool for building UI components.

React creates a VIRTUAL DOM in memory.

React finds out what changes have been made, and changes **only** what needs to be changed.

© Kindred Group plc 2021

Classified as General

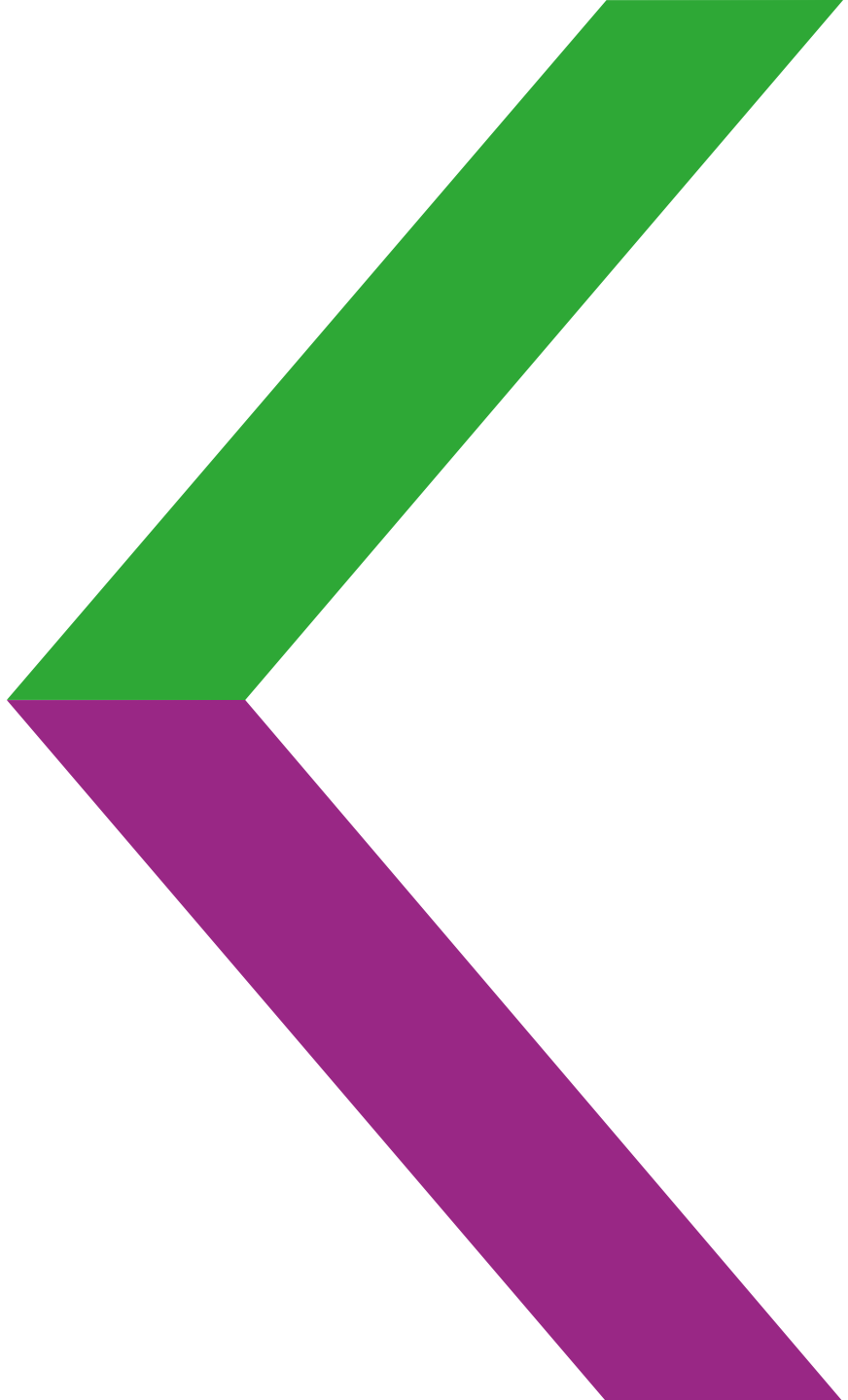
React to be continued…
session on February 10

© Kindred Group plc 2018

Home assignment:

Use HTML+CSS home assignment project and add logic to interactive elements.

Example: show alert and change the form background color after clicking ’submit button’.



© Kindred Group plc 2021

Free resources to read and practice:

https://www.w3schools.com/js

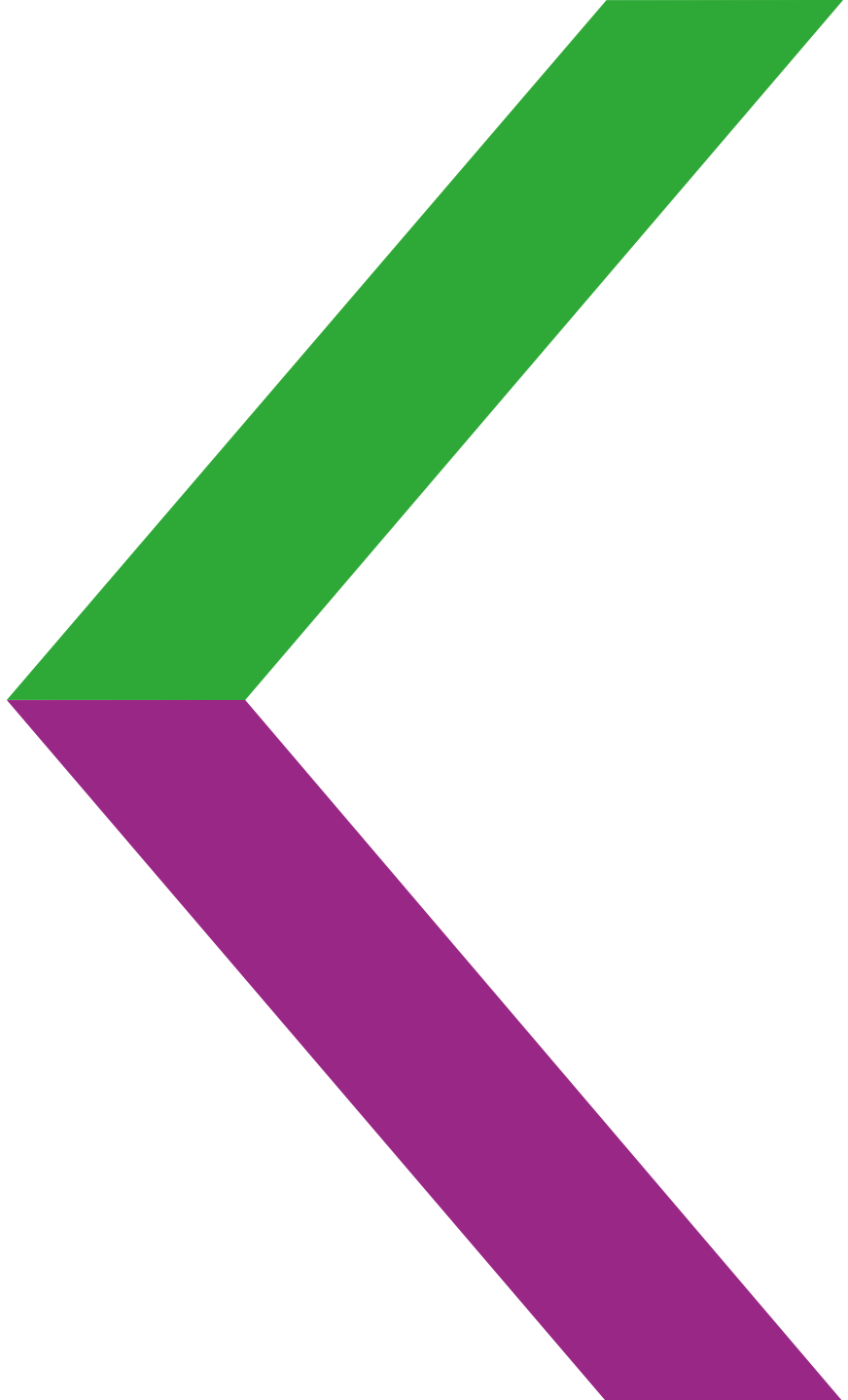
https://www.w3schools.com/REACT

https://reactjs.org/docs/getting-started.html

https://www.freecodecamp.org/learn

© Kindred Group plc 2018

Questions?



© Kindred Group plc 2021

Thank You

© Kindred Group plc 2021